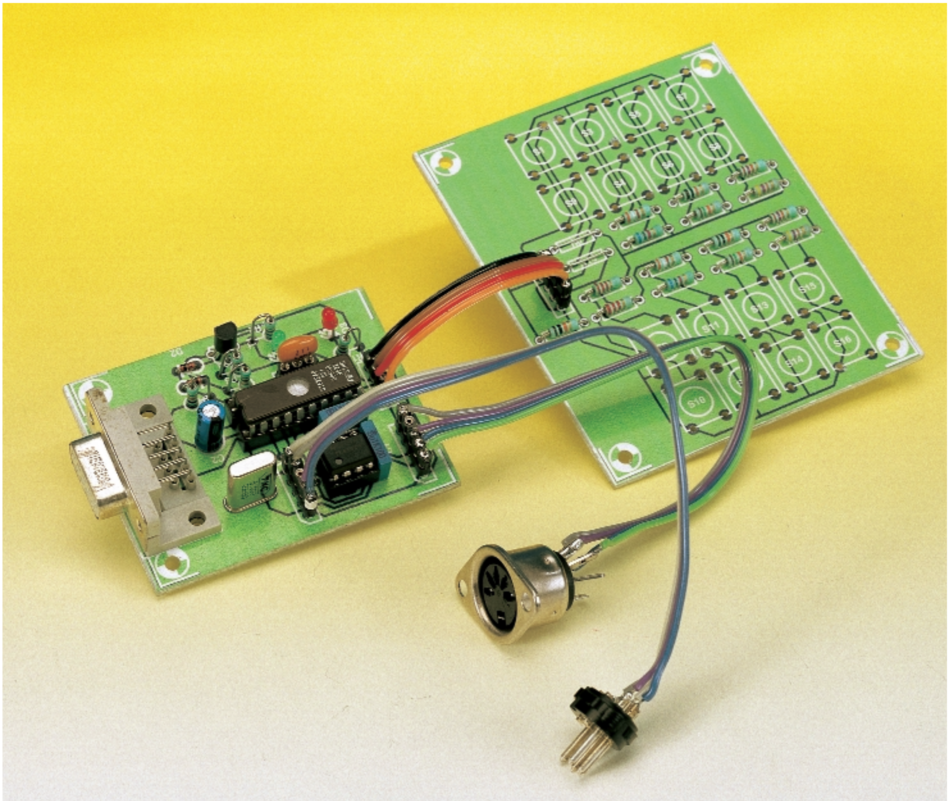
Hotkeys keyboard

part 2: construction, software and a few tips

Design by B. Stuurman

Last month we described the features and the operation of this handy extra keyboard. Now it's time to look at the hardware and the construction of the circuit.



The schematic diagram of the hotkeys circuit is shown in **Figure 6**. All pins of the connector for the PC keyboard (K1) are connected through to the connector for the PC (K4), with the qualification that there is a pair of analogue switches inserted in the clock and data lines. These allow the hotkeys keyboard to open these lines. Power for the circuit is taken from K4. In addition, you can see a connector for the RS-232 interface (K2). Resistor R5 and diodes D1 and D2 limit the incoming signal to levels of approximately 0 V and +5 V. The outgoing RS232 signal is directly connected to PB1 of the microcontroller. Although the signal levels here do not comply with the RS232 specification, they prove to work with nearly all PCs, and this simplified interface has been used in several *Elektor Electronics* designs.

A passive buzzer (Bz1) that is connected to port PB0 allows an acoustic feedback signal to be generated when a key is pressed, or in certain command situations. The 8-MHz ceramic resonator has built-in capacitors. Decoupling is provided by capacitors C1, C2 and C3.

The reset circuit is a little unusual. LED D4 is not used here as a light source (although it does emit light), but rather as a 'Zener' diode with a voltage of around 1.9 V. A real zener diode has too high a leakage current to function properly in this circuit. The LED and the voltage divider (R3 and R4) cause transistor T1 to start to conduct when the supply voltage reaches approximately 4.2 V, and to conduct fully at around 4.5 V. At a clock frequency of 8 MHz, correct operation of the microcontroller is guaranteed for a supply voltage of at least 4.5 V. Here we ensure that the microcontroller can never destroy the contents of the EEPROM by going amok due to an excessively low supply voltage. The voltage drop of the LED can vary from device to device, so it is recommended to check the supply voltage at which the reset signal is released. You can adjust the value of R3 if necessary.

Three outputs and three inputs are necessary for communication with the PC and the PC keyboard. Since port line PC4 cannot be used, as it is automatically connected to the Serial Peripheral Interface (SPI) of the microcontroller in the chosen configuration, two other inputs must be used for the connections to the hotkeys keyboard, namely PA3 and PC3. These are connected to K3, along with 0 V and +5 V.

Interfacing

There are so many connections in **Figure 6** that it is not readily apparent how communication takes place between the hotkeys keyboard and the PC, or between the hotkeys keyboard and the PC keyboard. **Figure 7** shows this much better. When line PB6 is a '1', the regular keyboard and the PC are connected to each other via analogue switches with a low on resistance (R_{DSon}) of around 60 Ω. If the hotkeys keyboard wants to send a string to the PC (due to a hotkey being pressed), PB6 is first set to a '0' level, which causes the switches to open. The microcontroller then looks for the string that belongs to the pressed hotkey, and it sends this to the PC a byte at a time using the 'send2pc' subroutine. This subroutine takes a byte from the buffer and sends it to the PC by switching PB2 and PB3, according to the transmission format described in Part 1 of this article. Before 'send2pc' actually sends a byte, it tests PA2. As long as this port is '0', the subroutine waits. Once the break string has also been sent (except for a free format string, when only the make string is sent), PB6 is again set to '1'.

PB6 is also set to '0' for the 'scankb' command. The microcontroller then changes PA0 from a normal input to an input with interrupt capability. As soon as the PC keyboard sends a code, it is intercepted by the interrupt routine. Before the interrupt routine exits, it sets status flags for 'scankb'. After the code has been received, PB7 is set to '0' and 'scankb' stores the received code in a separate buffer in RAM page 2. After this, PB7 is again set to '1'. If the buffer becomes full, due to a key being held down so that repeated codes are sent, an acoustic signal is emitted.

As long as 'scankb' is active, it is not possible to enter any commands from the PC keyboard. This mode can be exited by pressing hotkey 0.

In connection with the built-in Xmodem protocol, the RS232 interface must be 8 data bits wide. Although the Serial Peripheral Interface (SPI) of the microcontroller can receive an RS232 signal with 8 data bits, it cannot send such a signal. A separate routine called 'rs232out' is thus used for this. The SPI independently receives the RS232 input signal and generates an interrupt during the stop bit. The interrupt routine stores the received byte and sets a reception flag.

While the content of the EEPROM is being sent to the PC via Xmodem, the CRC checksum is accumulated during the transmission and send at the end. When a file is received, there is not enough time to directly

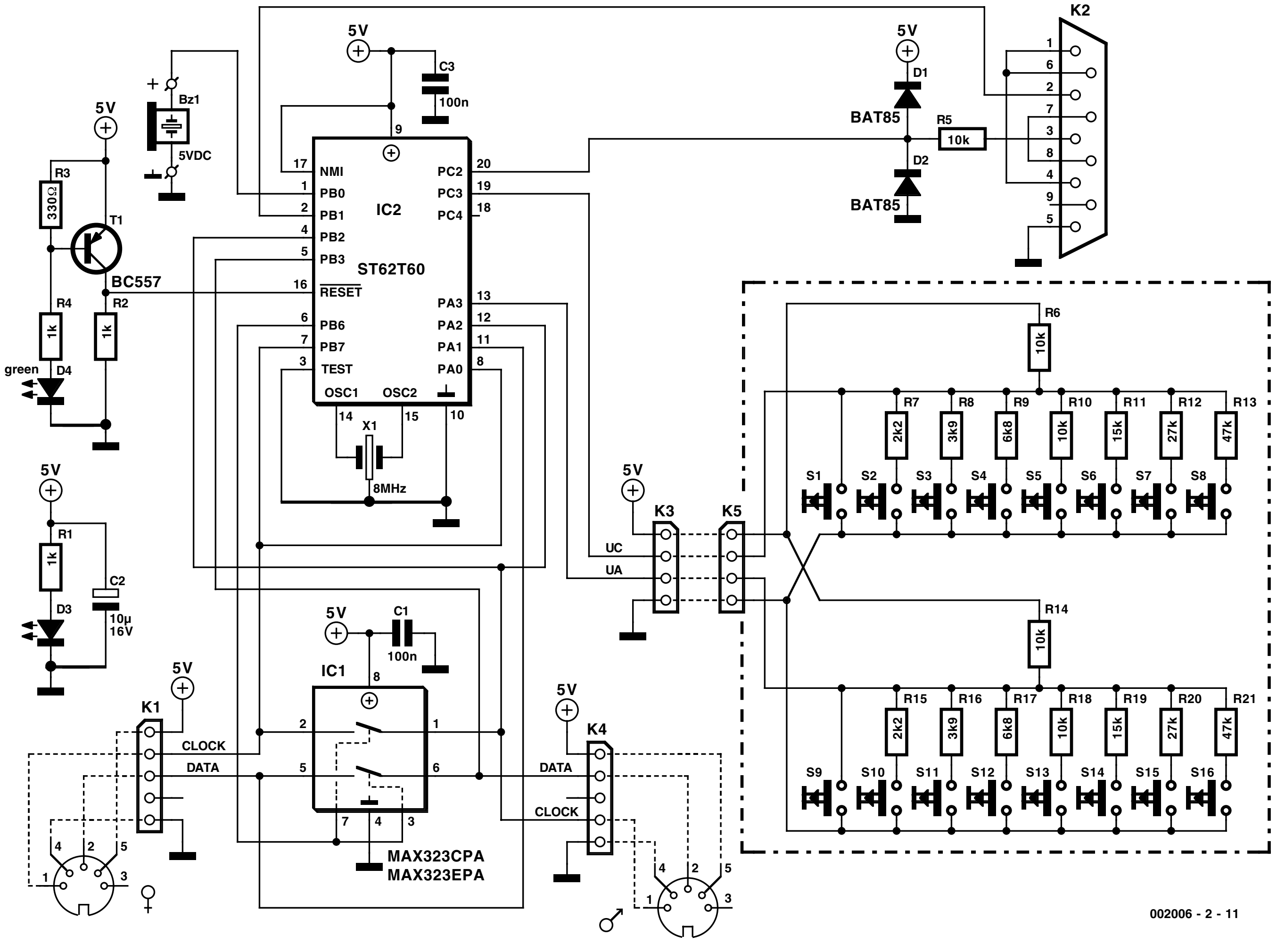


Figure 6. Schematic diagram of the complete circuit. The electronics does not amount to much more than a pair of ICs.

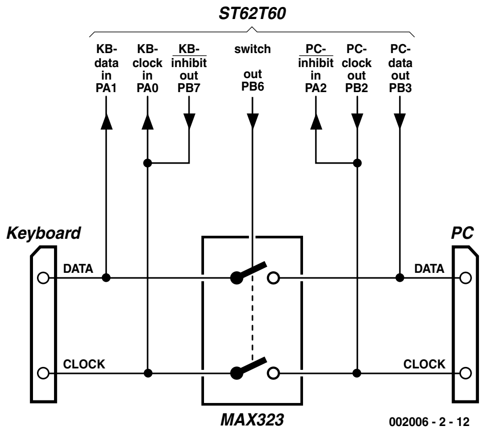


Figure 7. A clear illustration of the switchover mechanism.

Hotkeys behaviour

When you communicate with the hotkeys keyboard using Hyperterminal or some other terminal emulator program, the hotkeys may not behave exactly as you expect them to. This is not so surprising. When a key is pressed while Hyperterminal is running, the keypress is received by Hyperterminal as a keyboard key. After this, the keypress is sent to the hotkeys keyboard via the RS232 interface. The hotkeys keyboard stores the keypress in its buffer and then sends it via the RS232 interface back to Hyperterminal, which displays the associated character on the screen (if it is printable). This is why keys are not repeated, even if this option is enabled. It is thus recommended that you try out the hotkeys in the program in which you intend to use them.

store the received bytes in the EEPROM. RAM page 2 is thus used for temporary storage. However, this is only 64 bytes long, while an Xmodem data block is 128 bytes. Consequently, the remainder of the block is ignored after the first 64 bytes have been received, and these 64 bytes are then transferred to the EEPROM. After this, the microcontroller sends an error message, which causes the entire block to be sent again. This time data bytes 65 through 128 are extracted from the block and stored in RAM page 2. Once the entire block has been accounted for, the second part is also transferred to the EEPROM and the circuit sends an acknowledge signal in order to end the transfer. The CRC number is not tested.

The keyboard

We have already mentioned that two controller inputs (PA3 and PC3) are used for connecting to the keyboard portion of the circuit. To make things simple, we use the letter ‘A’ for everything relating to PA3 and the letter ‘C’ for everything relating to PC3.

It is very easy to connect a microcontroller input to the built-in analogue to digital converter (ADC). The ADC can measure from 00_H to FF_H, where 00_H corresponds to 0 V and FF_H to 5 V. The keyboard is divided into two groups of keys. The eight keys at the top of the schematic form the C group (left half of the keyboard), while the lower set of eight keys forms the A group (right half). Each key forms part of a voltage divider, so that a particular voltage corresponds to each pressed key. Resistor R6 is the upper resistor for the C group, and R14 is the upper resistor for the A group. The lower resistors for the C group are R7 through R13 (for keys S2 through S8), while the lower resistors for the A group are R15 through R21 (for keys S10 through S16). The keys are arranged such that S1 and S9 are next to each other on the lower left, while S2 and S10 are next to each other of the lower right. Since the two keys of each of these pairs of keys do not belong to the same group, it is always possible to measure whether S1, S9 or both are pressed, and likewise for S2 and S10.

The keys are scanned by the routine ‘getvolt’, with PA and PC alternatingly selected as the analogue input. The subroutine ‘getvolt’ can return a maximum value of ‘$8F_H$’, which means that no key is pressed. With the selected resistor values, the upper nibble matches the code of each of the pressed keys 1 through 8. In order to arrive at the correct key code for the A group, 8 must be added to his value to obtain 8_H through F_H (note that this means that keys 1, 2, 9 and 10 are special cases).

The ‘rtest’ command

Table 1 shows the 1% resistance values that can be used to obtain the most nearly ideal voltage values for the keys, in order to obtain the associated hexadecimal codes. It is however not necessary for the hexadecimal value to fall exactly in the middle of the range. The tolerance is so large that 5% resistors can be used, but in this case you should check whether the values of the keys are not critically close to the boundaries. The ‘rtest’ command allows this to be easily done. After this command has been activated, the hex value of each key of the chosen group (except for hotkey 1) is displayed on the screen when the key is pressed. The value of the lower nibble may lie between 4_H and A_H. If you type ‘rtest a’, group A will be displayed, while ‘rtest c’ looks at group C. For example, if you type ‘rtest c’ and then press hotkey 4, the displayed hex value should lie between 44_H and $4A_H$. If it is outside this range, the divider resistor belonging to this key must be adjusted, but this will not be necessary very often.

Table 1. Theoretical resistance values for the keys.

key	**S1**	**S2**	**S3**	**S4**	**S5**	**S6**	**S7**	**S8**
1% resistor value	0	2k00	3k92	6k49	10k2	16k2	26k1	49k9
potential (volt)	0.00	0.85	1.41	1.97	2.54	3.09	3.61	4.17
hex value	00	17	27	37	47	57	67	77
5% resistor value	0	2k2	3k9	6k8	10k	15k	27k	47k

open: 5.00 V = $8F_H$

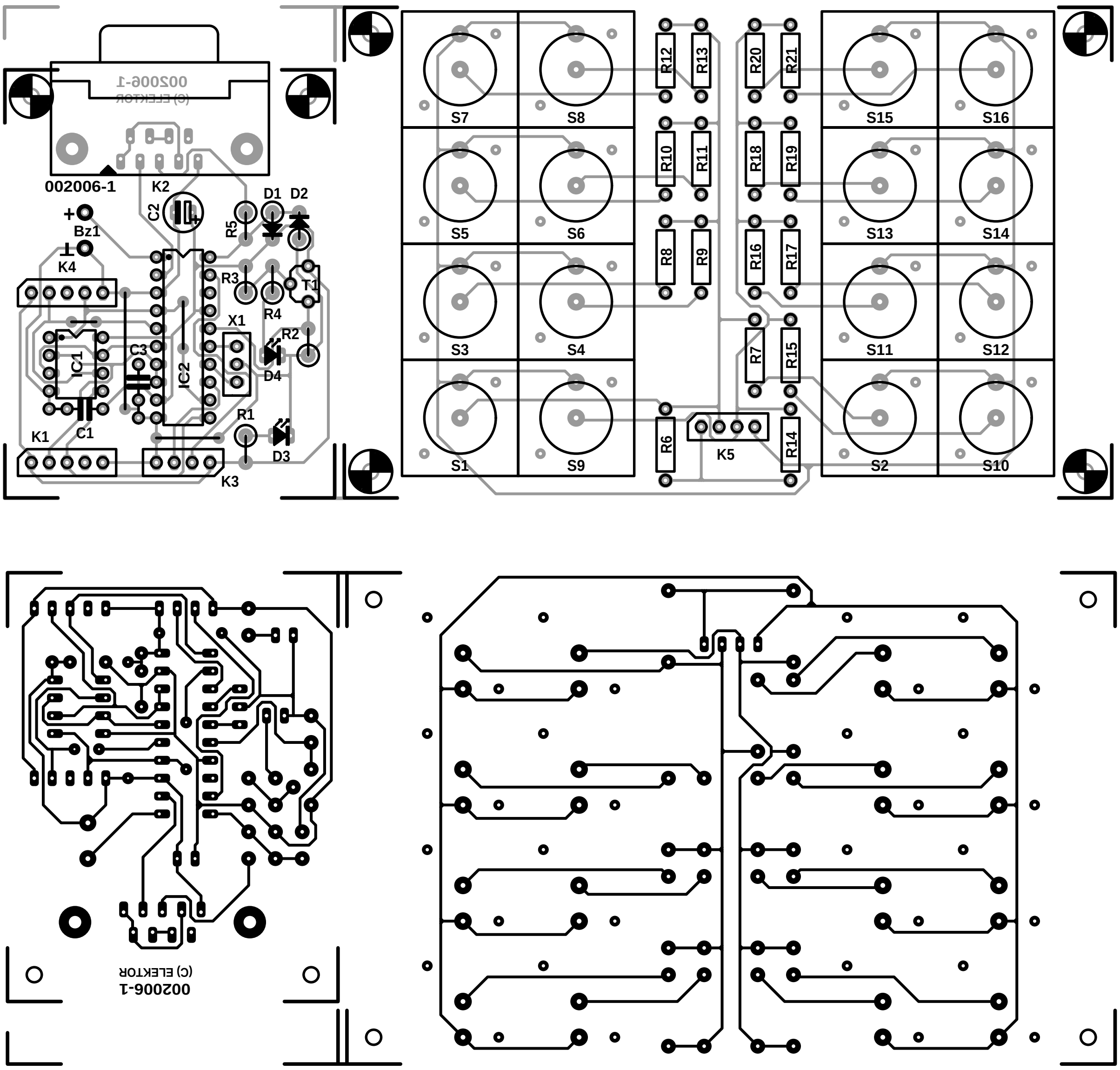


Figure 8. Circuit board track layout and component layout for the processor board and the keyboard.

Construction

The layout of the two circuit boards in shown in **Figure 8**, along with the associated component layouts. You can make the single-sided board yourself, but it is also available ready made through our Readers Services.

Start with assembling the processor board. As usual, mount the wire bridges first (four pieces). After this come the rest of the components, starting with the IC sockets and connectors K1, K3 and K4. These are lengths of board-mounted male headers that are available in strips. With K1 and K4, one pin is removed in order to ensure that the mating connectors can only be fitted in one way (after a pin has been inserted in the unused position). This is not necessary for K3, since the connections correspond to those on the hotkeys circuit board. Mount K2 after all the other components. Once this is done, you can press IC1 and IC2 into their sockets (make sure that IC2 is first programmed!).

For the keyboard, we have chosen Marquard keys in place of the more common (and less expensive) D6 keys. The size and touch of the Marquard keys are a much better match to the keys on a PC keyboard. They also contain real microswitches and are self-cleaning.

Now you have to make the three interconnecting cables. We used two lengths of 4-conductor screened cable for the connections to the PC and the PC keyboard. The cable to the PC is around 1.5 m long, while the one to the PC keyboard is around 50 cm long. Each cable has a mating DIN plug at one end and a 5-pin female circuit board header at the other end (and don't forget the polarisation pins). The pin assignments for the various cables can be seen in **Figure 6**. The keyboard circuit board is connected to the processor board by a 10-cm length of 4-way flat cable with a 4-pin male header at one end and a 4-pin female header at the other end. All wire connections to the header contacts can be covered with short pieces of heatshrink tubing.

The hotkeys keyboard can now be connected between the PC and the PC keyboard. When the PC is switched on, the hotkeys circuit emits a short beep, and the 'key name'

COMPONENTS LIST

Resistors:
R1,R2,R4 = 1kΩ
R3 = 330Ω
R5,R6,R10,R14,R18 = 10kΩ
R7,R15 = 2kΩ2
R8,R16 = 3kΩ9
R9,R17 = 6kΩ8
R11,R19 = 15kΩ
R12,R20 = 27kΩ
R13,R21 = 47kΩ

Capacitors:
C1,C3 = 100nF
C2 = 10µF 16V radial

Semiconductors:
D1,D2 = BAT85
D3 = LED, red, high efficiency
D4 = LED, green, high efficiency
T1 = BC557
IC1 = MAX323CPA or MAX323EPA
IC2 = ST62T60 (programmed, order code **002006-41**)

Miscellaneous:
K1,K4 = 5-way SIL pinheader
K2 = 9-way sub-D socket (female), PCB mount
K3,K5 = 4-way SIL pinheader
S1-S16 = pushbutton, e.g. Marquardt (Conrad Electronic o/n 70 68 92-55, with cap 70 08 60-55)
Bz1 = 5-V piezo-ceramic buzzer (passive)
X1 = 8 MHz ceramic resonator
5-way DIN plug, pins arranged at 180 degrees
5-way DIN socket, receptacles arranged at 180 degrees
10 cm 5-way flatcable
PCB, order code **002006-1** (see Readers Services pages)
Package: PCB and programmed controller, order code **002006-C**

must appear on the screen when a hotkey is pressed. In order to communicate with the circuit, you will have to connect the RS232 interfaces and run a terminal emulator program. The hotkeys can now be programmed, and the result can be stored as a file on the hard disk. A stored programming file can also be downloaded to the hotkeys keyboard, in order to assign a new set of functions to the keys.

A console-style enclosure (such as the Teko type 365) can be used to house the keyboard, or you can make your own enclosure from sheet aluminium.

(002006-2)

Design editing: Luc Lemmens.

Entering special characters

Many characters cannot be directly entered from the keyboard using a single key. Some familiar examples are the characters from 128 to 255 in the ASCII table. These are entered by holding down the Alt key and entering the decimal value using the numeric keypad. If you work with PostScript or TrueType characters, not all characters match the ASCII set (since Windows works with an ANSI character set). You can get a complete overview using the utility program ‘Charmap’, which also displays the 4-digit character codes. With some codes, the same numeral occurs twice in succession. Normally, this would mean that it is not possible to assign such a code to a hotkey (except as a free format string), since the key would have to be actually pressed and released twice (rather than repeated). The hotkeys output routine tests for double characters, and if they are present it first sends the break code of the first character before sending the make code of the second identical character.

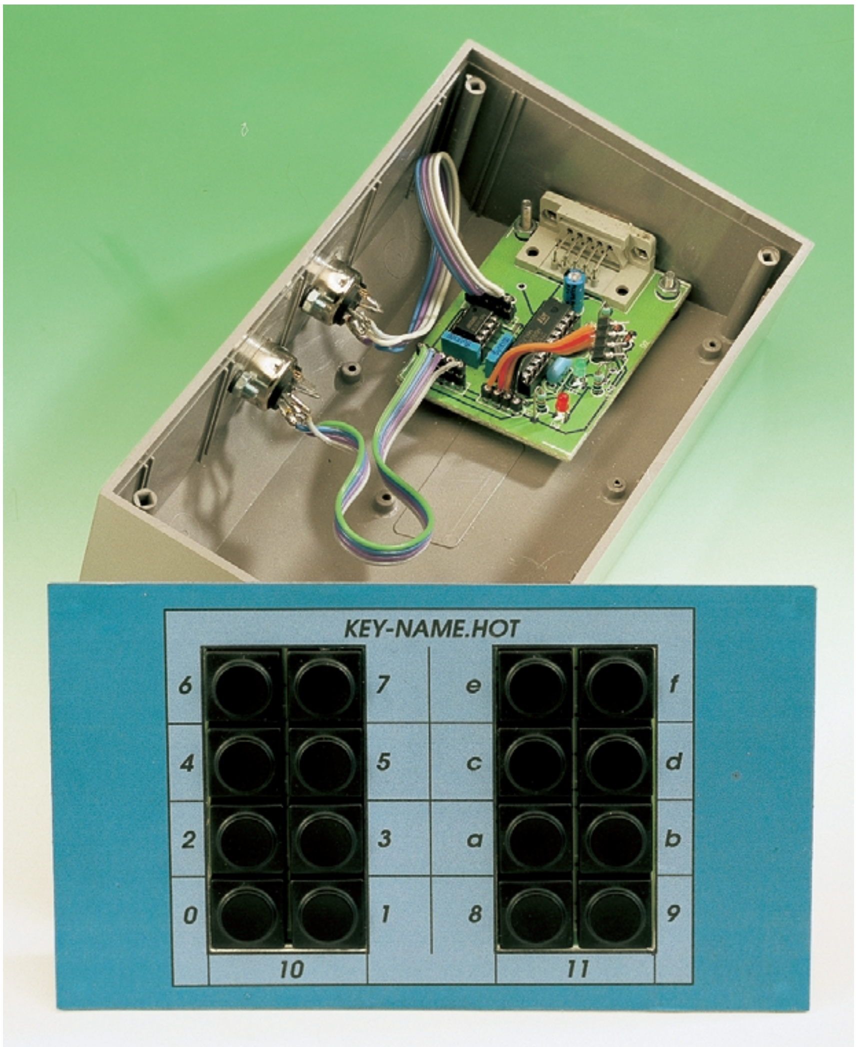


Figure 9. One way in which the boards can be fitted into an enclosure. If a metallic enclosure is used, the cable screens can be connected to the enclosure.